Dialogic®

Application Note

# Developing Media Solutions using RTSP and Dialogic® Products

## Executive Summary

This application note provides information about developing media solutions using the Real Time Streaming Protocol (RTSP) and Dialogic® Host Media Processing Software. Included (as a download) with this application note is a sample RTSP client library and a sample Dialogic® application, which serve as a reference for developing media streaming applications using RTSP.

## Table of Contents

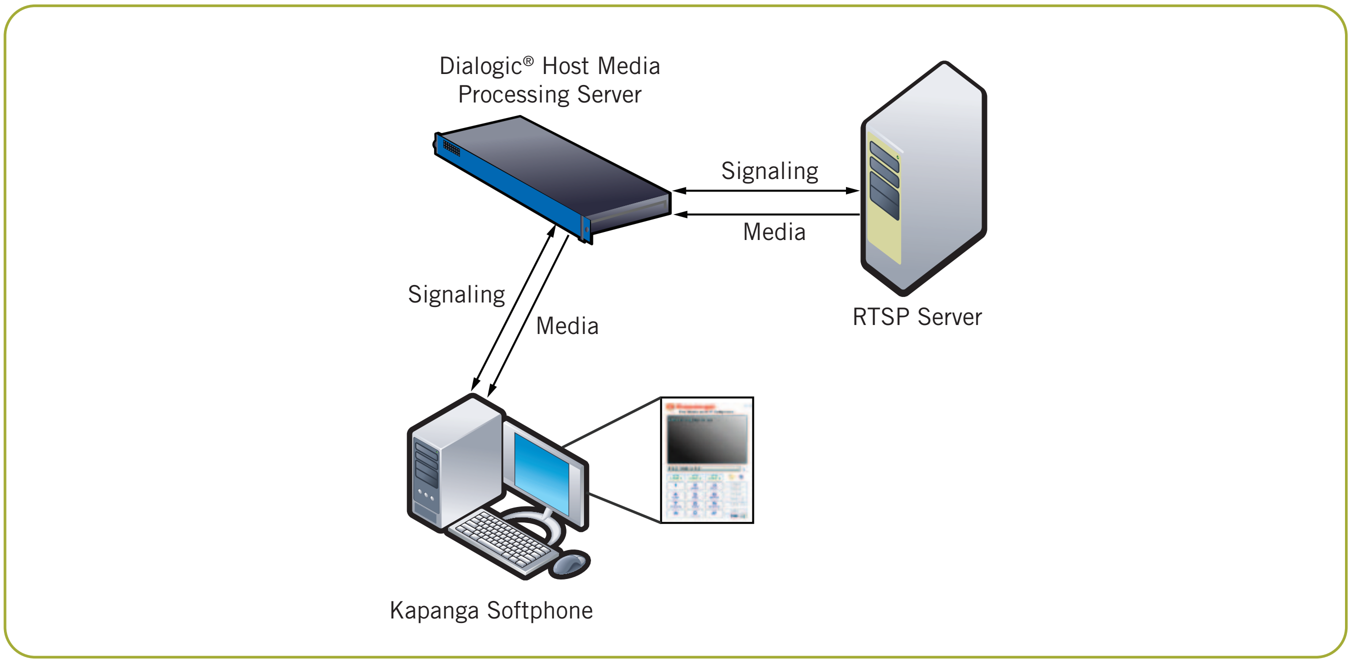

*Figure 1. VoIP Video Softphone Environnent*

## Introduction

RTSP (Real Time Streaming Protocol) is an Internet standard protocol commonly used for streaming media from a media server to a client endpoint. The RTSP client has the ability to control the media stream offered by the server with a series of VCR-type commands like *play, pause, fast forward,* and *rewind*. In addition to providing media to RTSP clients, RTSP servers maintain network status information, which is used to provide the best possible user experience.

This document provides information for consideration when building a Dialogic® Host Media Processing (HMP) Software application that uses RTSP to provide a media solution for both video-based VoIP phones and 3G and 4G cellular phones. The description of the architecture for both the VoIP phone and the 3G/4G solutions is followed by information about the sample application, plus code snippets.

A Zip file containing the source code for the RTSP client library and a sample Dialogic® application is available for download (see the *For More Information* section).

## Architecture

This section provides a complete list of the components used in building and running a Dialogic® RTSP-based media streaming solution. Specific information related to the components used in the sample implementation is provided. In addition, feature specifications of each component are provided so that they can be substituted with alternate products of the developer's choosing.

The components presented in Figure 1 are those used in the development of the sample application accompanying this document. Although the sample application has been used to stream media to 3G and 4G cellular devices, the components in Figure 1 provide a simple, inexpensive development environment.

### SIP User Agent

The Kapanga Softphone is a SIP softphone that is based on the Windows® operating system and that is used as the calling party in the sample application. Refer to the *Materials List for Sample Application* section for additional information. To provide the functionality for the sample application, the SIP user agent must support the media sent to it. The limitations on the coders used in the media that is sent depend on what the RTSP media server will serve and what a typical client can support. Commonly supported coders used to stream multimedia to a 3G or 4G endpoint are H263+ (also known as H.263/2000) for video and AMR Narrow Band (AMR-NB) for audio. H.263+ is common due to its ability to negotiate picture size, while AMR is common due to its being an adjustable rate coder. The Kapanga Softphone provides support for these coders, supporting both the

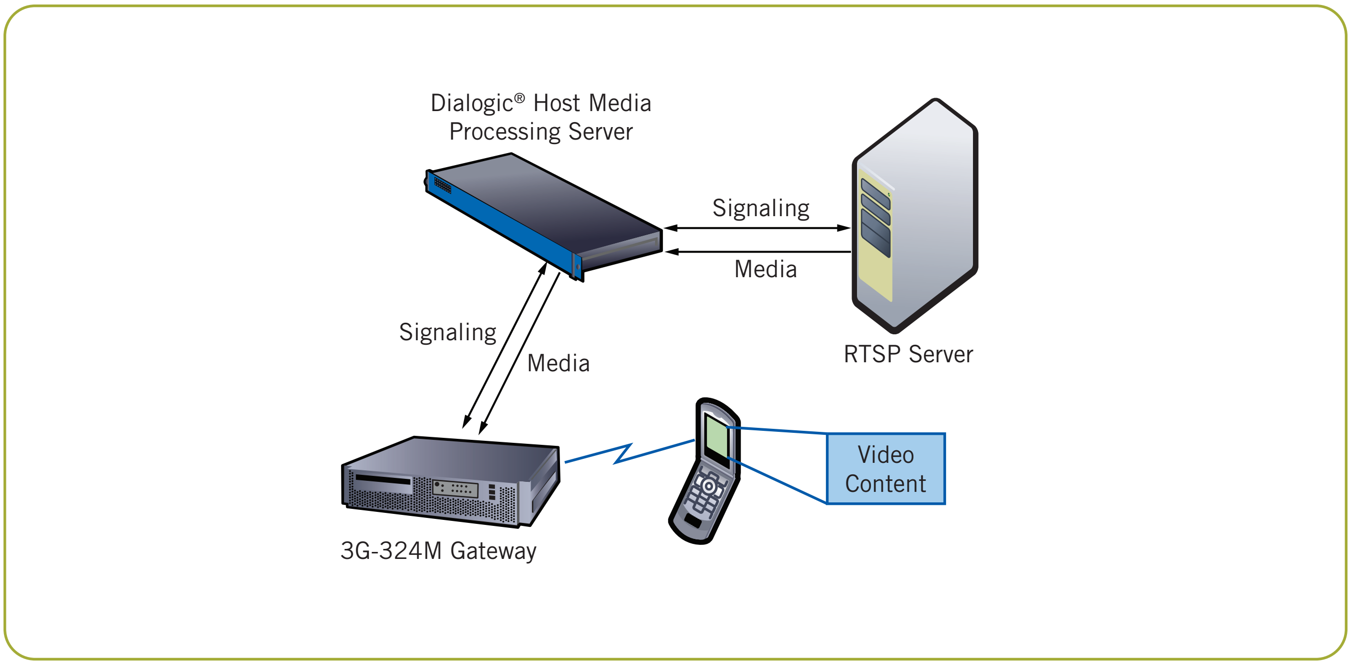

*Figure 2. 3G/4G RTSP Deployment*

RTSP server used and providing an alternative to the need for a complete 3G development environment. It is important to note that because the Kapanga Softphone terminates the media stream from the RTSP server, it must provide the coder licenses used to play the media received.

### Dialogic® HMP Software

Dialogic® Host Media Processing Software Release 3.0 for Windows® Service Update 155 was used due to its support for native hairpinning of both audio and video media. Native hairpinning (also referred to as tromboning) involves the routing of packets received on one media device to a second media device connected to the calling endpoint. The steps and the APIs used to accomplish native hairpinning are described in the *Code Walkthrough* section.

### RTSP Server

Apple's Darwin Streaming Server ("Darwin") is an open source RTSP server used as the streaming server in the sample application provided. Darwin was selected because it is open source, is easy to install and configure, and supports the media to be streamed. Media servers work on the concept of media containers. A media container is a combination of the video and audio content that is served to clients. The container provides information to the RTSP server on the format and characteristics of the media enclosed. Container types supported by Darwin include the MOV, 3GP, and MP4 formats, which provide the flexibility needed to serve media to both SIP user agents, as well as 3G and 4G cellular phones.

### 3G-324M Gateway

Figure 2 shows the components for adding support to the RTSP streaming sample application for 3G and 4G cellular devices. The 3G-324M Gateway is the added hardware component for providing RTSP streaming for 3G and 4G cellular users. Cellular users dialing into a media service will have their calls routed to a 3G-324M gateway, which serves as a PSTN to VoIP gateway. VoIP call signaling is then sent to the HMP server, which accepts the call and streams a menu of video choices back to the gateway for delivery to the caller's cellular phone. Streaming the menu to the caller was necessary due to the 3G-H324 gateway's inability to change coders during the course of the call.

## Code Walkthrough

This section details the steps used to implement the sample application. The coding steps are broken down into the topics covering the Dialogic HMP Software API and the RTSP client library calls.

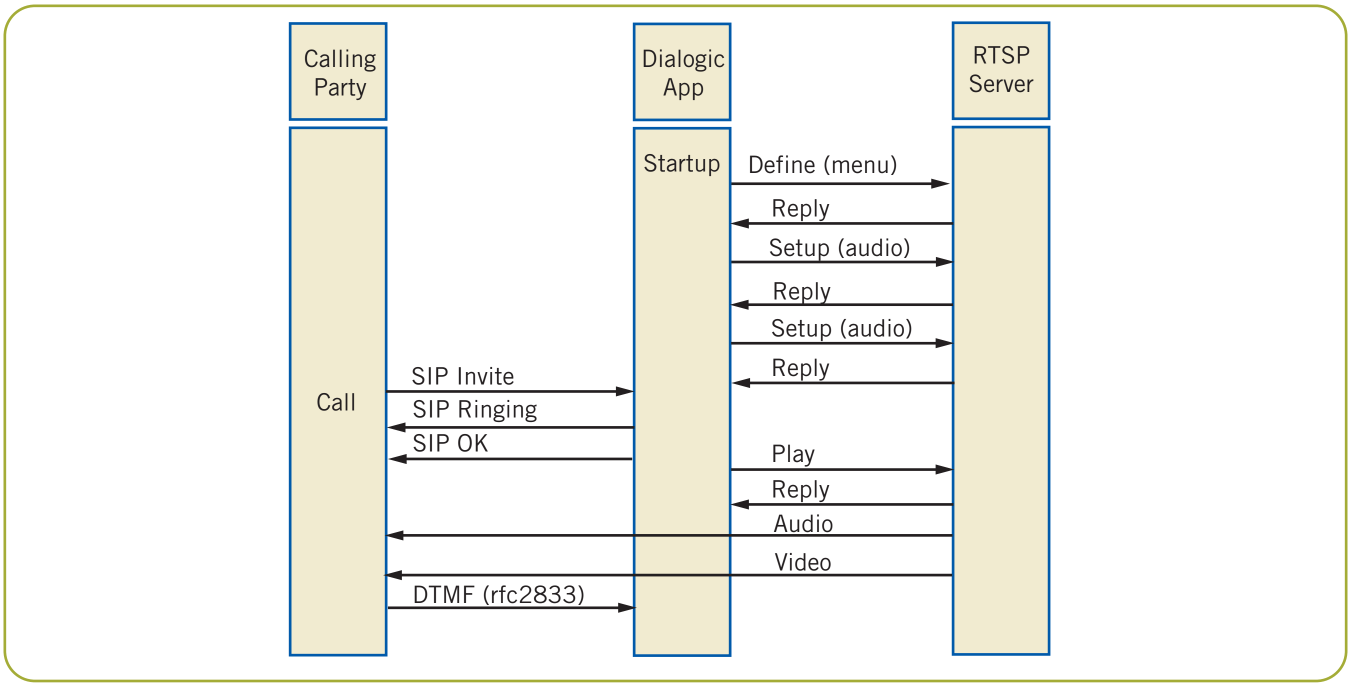

*Figure 3. Process Flow*

## Overview

Figure 3 presents the message exchange that occurs between the various parties involved in presenting RTSP-based media to the calling party. Figure 3 focuses solely on the messages exchanged between the calling party, the Dialogic® application, and the RTSP server. Messages exchanged between a 3G cellular phone and 3G-324M Gateway are outside of the scope of this document; however, for 3G solutions, the calling party shown in the figure would be the 3G-324M Gateway.

Figure 3 shows the message sequence for a Dialogic version of the RTSP application from startup through menu presentation and caller selection using the menu of options provided. When the DTMF digit shown as the last step in the figure is received, it results in an RTSP setup sequence identical to the first six steps that are shown here between the Dialogic application and the RTSP server. An RTSP Define request identifying the video the caller selected from the menu is sent. The RTSP reply contains the information needed to issue RTSP setup messages to support the video and audio streams. Replies received in response to the RTSP setup messages provide success or failure statuses for each Setup message sent. Once the last setup/reply message exchange occurs, the Dialogic application issues the RTSP Play function to start the media streams.

Figure 4 provides a summary of the Dialogic® APIs used and events received in an RTSP client application developed using Dialogic HMP Software. These Dialogic commands are used to establish the media hairpin for streaming media to the client endpoint. One device using Dialogic® Global Call Software, two media devices (IPM), and one voice device (VOX) are needed per channel to support the RTSP functionality described in this document.

## Using the RTSP Client Library

To assist developers in creating media streaming solutions, an RTSP client library implementation is provided along with the sample application. The source code provided is for consideration in developing an RTSP client library that provides the functionality to set up, play, pause, and tear down an RTSP media session. All commands are asynchronous and require a callback routine to process results returned. This code has been tested with the Darwin Streaming Server release 5.5.4. Developers may need to modify or replace the library if adding functionality or working with a different RTSP server.

**Note:** Variations exist in the signaling used to establish RTSP session by server implementation.

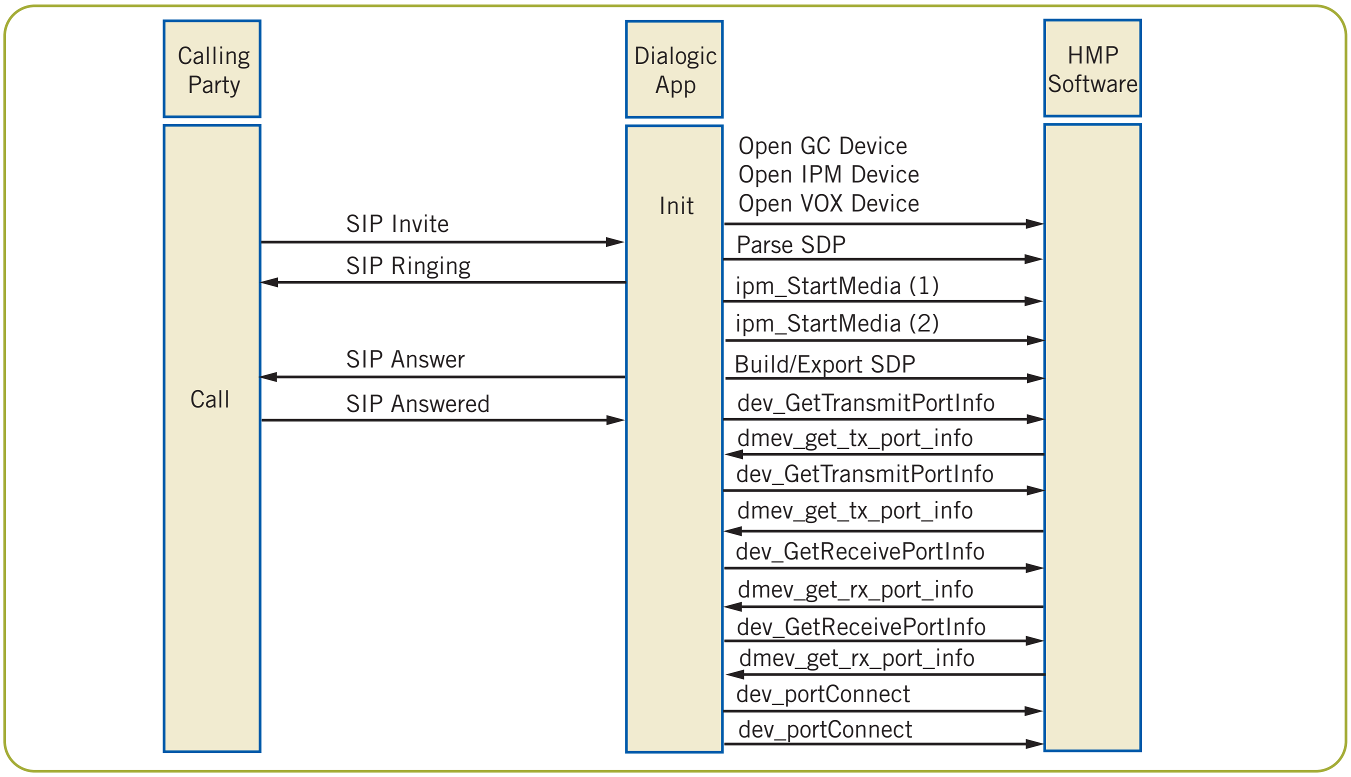

*Figure 4. Dialogic® API Flow*

### Startup

The setup process for an RTSP session may involve between six and eight message exchanges depending on the media requested. It is preferred that the Dialogic HMP Software-based application open one media session for each Dialogic HMP Software channel opened when the application is started. The RTSP *Play* function can then be issued when a call arrives on the channel, decreasing the time needed to present the menu to the caller.

The first steps in using the RTSP client library are to get a pointer to an instance of the library and then call its start method as follows:

```
// establish session with RTSP server for quicker menu presentation
CRtspSystem* g_rtspSystem = CRtspSystem::Instance();
g_rtspSystem->Start();
```

Next, the application needs to open a session with the RTSP client library for the device opened as follows:

```
if (m_rtspPtr->OpenSession(&m_rtspConfig) == 0)
      ::RegisterRequestByObject(m_rtspConfig.sessionHandle, this);
```

The configuration parameters represented by m_rtspConfig in the sample are both pre-configured and determined through the course of operation.

| Parameter | Value |
|---|---|
| clientAudioPort | Returned from IPM API call |
| clientVideoPort | Returned from IPM API call |
| callBack | Routine to receive asynchronous callbacks |
| Bandwidth | Amount of bandwidth requested for session |
| rtspServerAddr | Address of RTSP server—preconfigured |
| rtspServerPort | Socket number to receive RTSP signals—preconfigured |
| Language | HTTP specification for language (H14.4) |
| sessionHandle | Handle used to identify session for all RTSP library calls—returned value |
| mediaFile | Media file name to be streamed |

**clientAudioPort and clientVideoPort** — Contain the IP socket numbers that receive the audio and video streamed.

**callBack** — Contains a pointer to the application callback executed when a function completes. RTSP client library calls are asynchronous with the callBack parameter.

**rstpServerAddr and rtspServerPort** — Contain the IP address and socket number used by the RTSP server for signaling.

**Language** — Indicates the language used in the H14.4 HTTP format. U.S. English would be en-US, and case does matter.

**sessionHandle** — Contains the session handle value returned when the openSession call completes. This session handle must be used for future RTSP calls for this session.

**mediaFile** — Contains the name of the media file to be streamed from the RTSP server.

The openSession API starts the RTSP signaling process used to establish the session. RTSP signals generated from this API call include the RTSP Describe, followed by the RTSP reply/ok handshake needed to establish the RTSP media streams.

### *Play and Pause*

The Play API in the RTSP client library is used to start streaming the media to the Dialogic HMP Software-based application. Media received is hairpinned to the calling endpoint when received. The format of the Play API is as follows:

```
m_rtspPtr->Play(m_rtspConfig.sessionHandle, &m_playParms);
```

The parameters referred to as m_playParms include the following:

| Parameter | Value |
|---|---|
| cSequenceNumber | Unique value provided by the library |
| playLowRange | Starting point in media to begin play |
| playHiRange | Ending point in media to end play |
| preBufferMax | Maximum to buffer before playing |

**cSequenceNumber** — Receives the sequence number assigned by the RTSP client library and used by the RTSP server to track requests by client instance.

**playLowRange** — Indicates where in the media file to begin playing. To play the entire file, specify a value of zero.

**playHiRange** — Indicates how far into the media file to play.

**preBufferMax** — Specifies how many media buffers received are buffered before being played.

The Pause API is used to pause RTSP streams. RTSP client applications typically pause streams for a session before closing the session. The format of the Pause API is as follows:

```
m_rtspPtr->Pause(sessionHandle, &cSequenceNumber)
```

**sessionHandle** — Contains the session handle value returned when the session was opened.

**cSequenceNumber** — Receives the sequence number assigned by the library for the Pause function. This sequence number should be used in the callback routine in order to determine when the pause function has completed.

#### *CloseSession*

The CloseSession API generates the RTSP Teardown signal for ending the RTSP session with the server. All sessions should be closed in order to ensure that the streaming resources are freed for other applications or HMP ports. The format of the CloseSession API is as follows:

```
CloseSession(sessionHandle)
```

**sessionHandle** — Contains the session handle value returned when the OpenSession API was called to establish the RTSP session.

**Note:** All APIs in the RTSP client library return an integer value containing a 0 for success or -1 for failure. A value of 0 indicates the API function was submitted to the RTSP server for processing, while a value of -1 indicates the function failed and was not submitted to the RTSP server.

### Using the Dialogic® APIs

Dialogic HMP Software Service Update 155 was used in the development of the sample application accompanying this document. The Dialogic HMP Software feature used to support RTSP media streams is the native hairpinning functionality. Like other multimedia Dialogic applications, RTSP streaming applications use third party call control for all call signaling. Refer to the *For More Information* section for additional information on implementing third party call control.

#### *Third Party Call Control using Dialogic Global Call Software*

Although this document mostly covers the integration of RTSP with Dialogic applications, a brief overview of the use of the third party call control routines using Dialogic® Global Call Software as used in the sample application follows:

1. To use third party call control, the Global Call API library must be opened to specify its use. The StartGC function contains the code for opening the Global Call library specifying third party call control.
2. The SDP sent during the signaling for call setup must be parsed as the first step in responding to the inbound call request. The routine ParseOfferSDP in the sample application performs the parsing of the SDP information received with the SIP Invite message.
3. Once the SDP has been parsed, and information, such as video, audio ports and coders, have been processed, the next step is to populate SDP information to be used to respond to the SIP Invite. The BuildAnswerSDP function in the sample application builds the SDP message used.
4. After the response SDP information has been built, it must be exported for use in the signaling response. The ExportSDP function in the sample application exports the SDP and calls gc_util_insert_parm_ref_ex and gc_SetUserInfo to set the SDP for the call received.
5. A call to gc_AnswerCall is then made to complete the setup of the call.

The sample application provided could be enhanced to verify that the calling device had the coders needed to properly play the audio and video streams that were requested. As an example, verification could be accomplished via a database that contained a list of coders supported by caller or cell phone.

#### *Hairpinning Media Streams*

In a hairpinned application, media received on media ports connected to the source are routed to media ports connected to the destination endpoint. In this case, the media is received from the RTSP server, the source, and is sent to the calling endpoint, the destination. The Dialogic® devices used for the streams are media devices (IPM devices). The sample application uses one media device for the caller and one media device connected to the RTSP server. A network device using Dialogic Global Call Software is opened to receive call control and events. A walkthrough of the code used to accomplish hairpinning follows.

**Setting up Media Devices**

In the sample application, media devices are opened in pairs and are set up to receive DTMF using rfc2833. The ipm_Open and ipm_setParm APIs are used to open the device and set its DTMF mode to rfc2833. A call to ipm_GetLocalMediaInfo is made to get the video and audio ports associated with the media device. Both the video and audio ports are used in the hairpinning of the RTSP media stream.

DTMF digits collected from the caller are used to identify the media stream requested by the caller. Rfc2833 is used to ensure that DTMF digit events are generated for the application regardless of the audio coder used. A dx device is opened with each media device and a dx_listen is called using the media devices timeslot and is used to receive the DTMF events.

**Connecting Calling Endpoint to Media Stream**
The offered event is used to trigger the setup of the media devices for hairpinning the RTSP streams. The ipm_StartMedia is called for both the calling endpoint and the RTSP server media device. Information including video and audio coders, payload types, RTP and RTCP ports, frame size, and frames per packet is set prior to the ipm_StartMedia call. The sample application does not check the client endpoint's SDP to verify that it has the coders needed to play the RTSP audio and video streams. This verification step is useful for production implementations.

The answered event is then used to call the dev_GetTransmitPortInfo API. This API is called twice, once for the media device associated with the calling party and once for the media device associated with the RTSP server. These APIs are called asynchronously with the completion event, DMEV_GET_TX_PORT_INFO used to trigger the API calls to get the receive port information. Like the dev_GetTransmitPortInfo API usage, two calls are made asynchronously with the dev_GetReceivePortInfo API. One of the calls gets the receive port information for the calling party and the second gets the information for the RTSP server.

The asynchronous completion event generated when the dev_GetReceivePortInfo completes is used to start connecting the two media devices to facilitate the media stream. Transmit and receive information collected is placed into a DM_PORT_CONNECT_INFO structure, which is then used to begin connecting the two media devices. The API used to connect the media devices is the dev_PortConnect API and is called asynchronously. The completion event, DMEV_PORT_CONNECT is used to trigger the population of the second DM_PORT_CONNECT_INFO structure and to call to dev_PortConnect, which completes the media hairpin setup.

The event, DMEV_PORT_CONNECT, generated when the second call to dev_PortConnect completes, is used to trigger the RTSP library Play request. Successful execution of this request results in the media stream being streamed from the RTSP server through the hairpinned media connections to the caller's endpoint.

**Note:** All API commands listed in this section, with the exception of the ipm_Open function, must be made asynchronously.

**Disconnecting Media Streams**
The disconnect event GCEV_DISCONNECTED should be used to trigger:

1. The pausing of the RTSP media.
2. The disconnection of the media device hairpin. Like the other dev API calls, dev_PortDisconnect is called asynchronously, and like dev_PortConnect, it must be called once for each of the connected media devices.

Only one call to dev_PortDisconnect for a connected session can be processed at once. The completion event, DMEV_DISCONNECT, should be used as a trigger to additional calls to dev_PortDisconnect. Failure to use the dev_PortDisconnect API may result in error conditions occurring with future calls to connect the media devices.

**Note:** Although the sample application architecture has a one-to-one mapping of callers to RTSP media sessions, it is possible to add callers to an existing RTSP media stream following the steps presented in this document. An application of this technique would be to add callers into a live streaming event. Adding callers to an existing RTSP stream is achieved by connecting the caller's media device to the RTSP media stream via a dev_portConnectCommand.

**Application Cleanup**
All devices and the RTSP session(s) should be properly terminated and closed before the application exits. For RTSP, a call to the CloseSession function is required for each open RTSP session. CloseSession results in a RTSP Teardown call being made to the RTSP server for the session freeing the RTSP server resources. The dx devices

associated with the media devices should call dx_unlisten and then close, as should the media devices. Finally, the network devices using Global Call Software should be closed and the Global Call API library stopped.

## Materials List for Sample Application

Kapanga Softphone version 1.00.2158c — Ecotronics Ventures LLC

Darwin Streaming Server version 5.5.4 — (will need to set up an account to download) http://developer.apple.com/darwin/projects/streaming

Dialogic® Host Media Processing Software Release 3.0 for Windows® Service Update 155

Network protocol analyzer (Ethereal or Wireshark [formerly known as Ethereal])

## Appendix A: Setup and Configuration of Darwin Streaming (RTSP) Server

The Darwin Streaming Server can be downloaded from http://developer.apple.com/opensource/server/streaming/index.html.

1. Download the Darwin Streaming Server as a self-extracting .zip file named DarwinStreamingSrvr5.5.4-Windows.exe. Running this file prompts for a directory into which to unzip the contents.
2. From the directory containing the unzipped Darwin distribution, run the install.bat file.
3. Darwin's install.bat moves the files it needs to run to the root:\program files\Darwin Streaming Server directory structure. Where root: equals the root drive on the server and is usually c:. Additionally, install.bat adds the Darwin Streaming Server as a system service on the server.

By default, Darwin uses Reliable UDP (rfc908 and rfc1151) as the delivery transport for media. Reliable UDP acts much like TCP in that acks are exchanged for each UDP packet sent. Not responding to a fixed number of packets results in the media session being closed by the Darwin server. Reliable UDP is not supported in the Dialogic® product set and is not widely supported in telephony-based endpoints. You can turn off Reliable UDP by editing the streamingserver.xml file changing the following line:

```
<PERF NAME=”reliable_udp”
TYPE=”Bool16”>true</PERF>
```

To:

```
<PERF NAME=”reliable_udp”
TYPE=”Bool16”>false</PERF>
```

Stop and restart the Darwin Streaming Server to remove the use of Reliable UDP for all media streams. The Darwin Streaming Server works without any manual intervention on the server. By default, the Darwin Streaming Server logs are written to the root:\ Program Files\Darwin Streaming Server\Logs directory, and are a good source of information for troubleshooting streaming issues.

## Acronyms

| | |
|---|---|
| **3GP** | 3GP is the media container frequently used by mobiles, |
| **AMR** | Adaptive Multi-Rate — Is a variable rate audio compression coder that is used both in the 3G Cellular and VoIP worlds. As more congestion is detected on the network, AMR lowers the coder sample rate. |
| **AMR-NB** | Adaptive Multi-Rate Narrow Band — Provides sampling rates between 4.75 kbps and 12.2 kbps. Audio sent is mono. |
| **Hairpinning/ Tromboning** | Information/data or media going into a switch and turning around and going back to another device. |
| **MP4** | The MPEG 4 media container. |
| **MOV** | MOV is the Apple QuickTime media container. |
| **RTSP** | Real Time Streaming Protocol — Application-level protocol used to control the delivery of data with real time properties such as video and audio. |
| **Reliable UDP** | Reliable User Datagram Protocol is a simple packet-based transport protocol, based on rfc908 (version 1) and rfc1151 (version 2), which is intended as a reliable protocol to transport telephony signaling across IP networks. |

## For More Information

*Adding video to an R4 IVR Application* — On-line training material — http://www.dialogic.com/support/training/

*Global Call IP for Host Media Processing Technology Guide* – Chapter 5 — http://www.dialogic.com/manuals/docs/globalcall_for_ip_hmp_v8.pdf

Real Time Streaming Protocol (rfc2326) — http://www.ietf.org/rfc/rfc2326.txt

A Zip file containing the source code for the RTSP client library and a sample application can be downloaded at http://www.dialogic.com/goto/?10584

To learn more, visit our site on the World Wide Web at **http://www.dialogic.com**.

**Dialogic Corporation**
9800 Cavendish Blvd., 5th floor
Montreal, Quebec
CANADA H4M 2V9